Linear Programming and its Applications to Computer Science

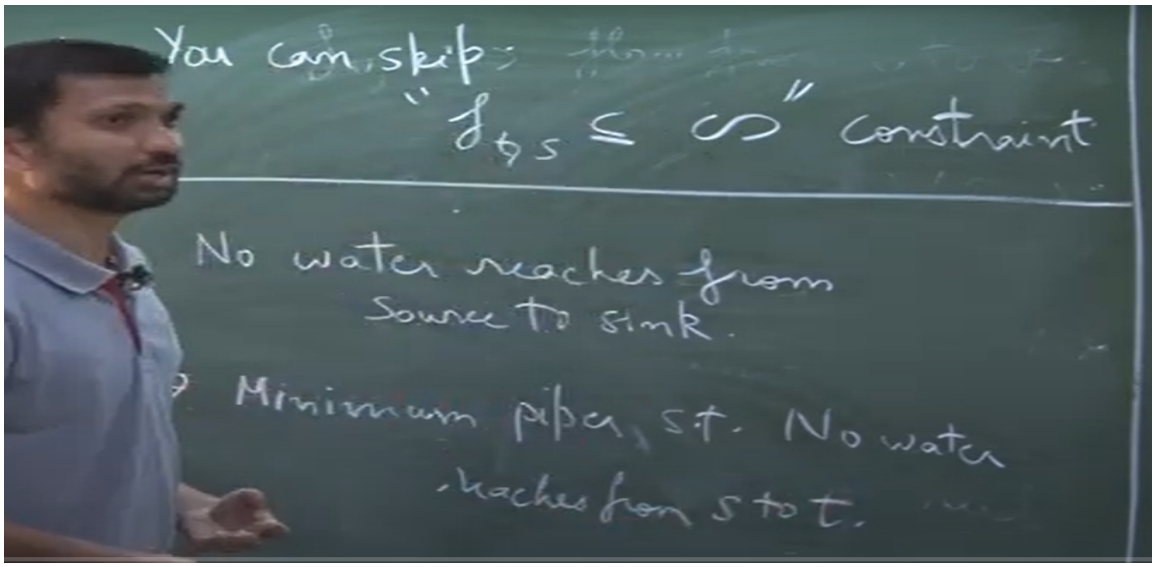
Prof. Rajat Mittal
Department of Computer Science and Engineering
Indian Institute Of Technology, Kanpur

Lecture – 40
LP for Min-Cut Problem

And now, we will look at a very different problem. In this case, you are the nice guy.

You have to provide water from source to the sink, correct. But nice guys are not fun, right. Fun happens when you try to destroy things or ok joking ok. Here our life will become interesting if we say how can I make sure that no water reaches from source to the city, ok.

Once again assume you are a terrorist or you know whatever terrorist for one you know or spy for another country or whatever and you want to make sure that from correct and obviously all bad guys are lazy. So, if you want to stop getting everything from here to here, one simple strategy is cut all the pipes right. But that is not great. That is expensive or we are lazy or we do not have resources to cut all the pipes, right.



In that case, you want to cut minimum pipes such that no water reaches yes no water reaches from as they were saying I want to cut minimum number of pipes such that I want to sum up the capacities of those pipes. Why am I summing up the capacities? Because intuitively I can assume Cu,v is the amount of effort to cut pipe right. If the capacity is big pipe is going to be bigger I need more resources agreed. This is called the Min-Cut Problem. I should not have erase the problem but. So, something like this right and then what does it mean? How do I make sure that if I what does it mean that if I cut some pipes no water reaches from S to T.

What does it mean mathematically? Can I describe it as a nice condition? There is a there is path or there is no path right. So, if from S to T I can cut it means there is no path from S to T. Can I make it more precise? Can I simplify it because this path there are so many paths right. Any time I cut the pipes I will have vertices which are still connected to S and vertices which are not connected to S right. So, cut means I cut some edges every cut specifies S and S bar where S is the set of vertices connected with S agreed right set of vertices connected with S and correct.

I want to make sure that S and T I have no water going from S and T that means what do I need right. So and what should be part of what and what should not be part of what right. So, this is the definition of a valid cut. So I can take any subset of my vertices which contains small s, but not T then whatever edges are going from S to S prime is the capacity of the cut. So, now we have another problem we have for given a vertex at S and T in defined valid cuts.

Cuts are specified by a subset of vertex which contains S, but does not contain T right. And given this cut S I do not even have to specify S bar given this cut S this is the capacity of the cut. And then the problem of the terrorist is to minimize the cut agreed.

And now if you believe in the beauty of mathematics and probably linear programming you have a problem here you have a problem here what do you anticipate. You have to wait that this is the let us try.

So why do not you write this as a linear program do not take the dual of this. As a mathematical problem let us write down the let us write down the program for this an optimization program. Hopefully a linear if it has to be dual then it has to be a linear program right. So, and if you realize we did something like this before right in the exam we had matching and covering and we cut we made a program and then converted into a linear program. So, what do you think what would be the you start with a program for this what should be the variables what should be the equations.

So, if you cannot come up with a program where are you stuck right. That is how we will define their values remember like maximum matching what did we say we said whatever is part of matching we will call it 1 correct. And whatever edge is not part of the matching we will call it zero right. So, let us start with that let us say p u is the variable I am just not writing it as x u. So, that forum has to change everything and change all the variables.

So, p u is indicator of whether u is element of s or not correct make sense this is the simplest thing which will be. So, helpful for you all the time this indicator variables right what it means is just to clarify this is takes 2 values 1 and 0 it indicates when it is 1 it is 1 if u is element of s 0 otherwise this is called an indicator variable you can replace this by any proposition or any formula right. So, anything which can take a true or false value. So, never this is true variable takes value 1 otherwise 0 like in matching x was part of the matching correct. So, let us examine the word cover otherwise 0 right.

So, this is suppose p u is this sufficient what is what is what is the thing which I want to optimize summation p u c u v good is this the correct. This capacity will only be counted when p u is 1 and p v is 0 sorry minus into. This is a problem right why is this a problem not linear right you are multiplying p u by p v what about this yes sometimes it will be 0 Are you saying this? Obviously the issue is reverse edges will start counting here we only want 1 side edges right. So, it will subtract the capacities of edges going from s bar to s right which we do not want we want to show dual we cannot even write a program for it. And I will give you the solution because I know forum has a solution the idea is exactly this is indicator u v is part of the cut and remember how we define the cut edges going from s to s bar.

If we do that then first thing becomes easy correct this is not, but now what are we stuck with validity right it should happen that if p u is 1 and p v is 0. Similarly if p u is 0 p v is

0 then d u v should be 0. So, how do I enforce that constraint d u v plus once again anyone who does not have a pen and paper. So here the first try could be this almost works correctly except right. So, this almost works correctly except when u is element of s bar and v is element of s then you will have infeasibility problem right like you will have some u v's where this is 0 this is 1.

And if you are saying enforce the condition d u v greater than equal to 0 that we will do, but if we just do this then we will have no feasible solution because for any sensible cut or we will only have trivial solutions. Any sensible cut will have somewhere this will become 0 this will become 1 I cannot find a d u v for this and this is where linear programming actually saves us why because what are we trying to do trying to minimize d u v right. So if p u minus p v is 1 minus 1 to be 1 when this is 0 since we are minimizing this will automatically make it 0 for the other direction when this was minus 1 this constraint will again force it to be 0 because we will have this constraint. Take some time to think about it, but there is a nice little trick which can be used basically because we are working in this optimization domain. We are trying to optimize in any sensible or any good solution of this we are anyway trying to optimize we are looking at the optimal solutions there might be some frivolous solutions, but the useful solutions will make sure whenever this is 1 this will be 1 whenever this is 0 this will be 0 whenever this is less than 0 this will remain 0 make sense because we want to minimize this.

By the way we also want so I will just write this constraint explicitly right. But that would not that will be counted here that would not be I do not want to erase this right. So are you with this change the only thing here now is what where I am at constraining P U P V D U V to be correct at this point if I want to exactly correspond with the cut I want P U as well as D U V to be element of 0 comma 1. Notice the curly brackets here this is an indicator variable. So, I say that I constraint them to be between 0 and 1 sorry not between 0 and 1 they should be exactly 0 or 1.

So what you want to convince yourself is that the optimal solution here will be the main cut solution. There is a small change here, but then if you look at any solution here correct what will happen only D U V is which are 1 will be the case when P U is 1 and P V is 0 right. So then I define my cut to be wherever P U is 1 and then that will be a valid cut because I will keep D U V to be 1 only and only if this is in S this is not in S. So this is what you want to be sure of. So, let us say that is saying if and only if P U is equal to 1.

Sounds good. So we are very happy we have a linear program for max flow problem. We have a integer linear program for the min cut problem. So clearly they cannot be dual

of each other. So before I talk about the relation between these problems. I want to talk about ILPs or integer linear programs.

These are almost linear programs where some variables are constraint to be integers and clearly ILP consumes LP right because I am allowed to put 1 more constraint. I can put linear constraint on my variables as well as I can say that my variable should be integers right. So now you remember in the nice. Lint program was nice because it captures lot of problems and it has efficient solution. It has good algorithms to solve it right.

Simplex was 1. It was not efficient, but was a nice algorithm and we know nice algorithms exist. So now another question could be what about ILPs? For LPs we have nice algorithms. Do ILPs also have nice algorithms? What do you think? Why? No, you have to expand your solution approximately. No no no you are saying how LPs are useful, but do you believe ILPs are efficiently solvable? Yes or no? So what is the argument for no? Good. So intuitively our feasible set was convex like nice continuous one right.

Now once you force it to be integers the niceness of our feasible set has been lost right. You can say it in multiple ways, but more importantly this is just an evidence in some sense, but actually we have a much concrete evidence that this is very hard because many NP hard problems can be written as integer programs. And if you are familiar with some you can take the max independent set problem and write it as an integer programs. And we obviously not obviously, but most of us believe that NP hard problems cannot be solved in polynomial time right. Writing up the solution is much harder than verifying the solution.

And if that is the case that means ILPs in general cannot be solved in polynomial time. So other way to say it is NP not equal to P implies ILPs cannot be solved efficiently right. One thing which I want to emphasize here is that when I make such an statement I am saying this about the generic class. It could be specific ILPs which I can still solve this is going to be an example of that. So just because something is ILP does not imply that it is hard.

What we know just because it is an ILP does not mean it is easy. It might be easy it might be difficult. Do I make myself clear on this right good. So with this knowledge now we are stuck here. We have D prime which is an ILP right.

And we do not know whether it is easy to solve or hard to solve. But we have a trick a parsley. Remember the trick which we did in the exam also. What is the trick? Yes, we make it a linear program and ask does it change the value. So the trick is let us make it a

linear program.

Let us change it to the interval 0, 1. First good thing about this is that this is a linear program right. That is a great news. The bad thing about this is the solution here could actually be it could be non-integer. But what is its relation with the min-cut problem? The solution of this smaller right because this is going to have many more solutions as compared to the ILP right.

This is going to be smaller than the min-cut the actual min-cut. Question is can we show that they are close? This is where your suggestion of you know sometimes we can take these solutions make them integer and not lose much. This thing is called rounding we will study about it. What I am trying to say is this is a generic technique.

This is a special case. Here things are going to be wonderful that does not happen generally. We have an integer linear program. We convert into a linear program. We generally want to say that they approximate each other well.

Even that is not true in general. Mostly the way we show that the approximate each other well is you take the solution of let us say this d. This has some fractions involved. You want to convert them into integers and still keep the feasibility. This is called rounding.

It looks like rounding right. We say if it is 0.3 let us make it 0. If it is 0.9 let us make it 1. After that we have to say that our solution is feasible and we have not lost much in the objective.

We will see many examples of such rounding techniques. This is a standard technique where we convert an ILP into LP by making the variables continuous. Sometimes we do not lose anything. Sometimes we lose a bit.

Sometimes we do not get any guarantees. But anytime we are stuck with an integer linear program it is worth converting into an LP and asking is this LP a good enough approximation of my original ILP.